

13

Fundamentals of NumPy

聊聊 NumPy

本节的核心是用 NumPy 产生不同类型数组



重要的不是生命的长度，而是深度。

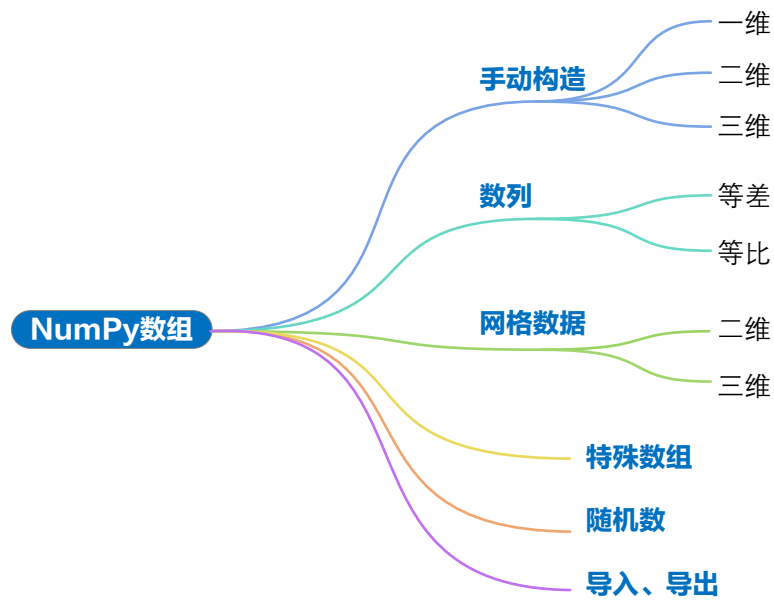
It is not the length of life, but the depth.

—— 拉尔夫·沃尔多·爱默生 (Ralph Waldo Emerson) | 美国思想家、文学家 | 1942 ~ 2018



- ◀ `math.ceil()` 向上取整
- ◀ `matplotlib.cm` 是 Matplotlib 中的一个模块，用于颜色映射
- ◀ `matplotlib.patches.Circle()` 创建正圆图形
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `matplotlib.pyplot.scatter()` 绘制散点图
- ◀ `numpy.arange()` 根据指定的范围以及步长，生成一个等差数组
- ◀ `numpy.array()` 创建 array 数据类型
- ◀ `numpy.empty()` 创建指定形状 NumPy 空（未初始化）数组
- ◀ `numpy.empty_like()` 创建一个与给定输入数组具有相同形状的未初始化数组
- ◀ `numpy.exp()` 计算括号中元素的自然指数
- ◀ `numpy.eye()` 用于创建单位矩阵
- ◀ `numpy.full()` 创建一个指定形状且所有元素值相同的数组
- ◀ `numpy.full_like()` 创建一个与给定输入数组具有相同形状且所有元素值相同的数组
- ◀ `numpy.linspace()` 在指定的间隔内，返回固定步长等差数列
- ◀ `numpy.logspace()` 创建在对数尺度上均匀分布的数组
- ◀ `numpy.meshgrid()` 创建网格化坐标数据
- ◀ `numpy.ones_like()` 用来生成和输入矩阵形状相同的全 1 矩阵
- ◀ `numpy.random.multivariate_normal()` 用于生成多元正态分布的随机样本
- ◀ `numpy.random.uniform()` 产生满足连续均匀分布的随机数
- ◀ `numpy.zeros()` 返回给定形状和类型的新数组，用零填充
- ◀ `numpy.zeros_like()` 用来生成和输入矩阵形状相同的零矩阵
- ◀ `seaborn.heatmap()` 绘制热图





本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

13.1 什么是 NumPy?

简单来说，NumPy 是 Python 科学计算中非常重要的一个库，它提供了快速、高效的多维数组对象及其操作方法，是众多其他科学计算库的基础。下面展开聊聊 NumPy 的主要功能。

NumPy 最重要的功能之一是提供了高效的多维数组对象 `ndarray`，可以用来表示向量、矩阵和更高维的数组。它是 Python 中最重要的科学计算数据结构，支持广泛的数值运算和数学函数操作。

此外，如果大家需要处理有标签、多维数组数据的话，推荐使用 `Xarray`。`Xarray` 可以看作是在 `ndarray` 的基础上，增加了标签和元数据的功能。`Xarray` 可以对多个数组进行向量化计算，避免了循环操作，提高了计算效率。`Xarray` 提供了多种统计分析函数，可以方便地对多维数组数据进行统计分析。本书不会展开讲解 `Xarray`。

NumPy 提供了多种数组操作方法，包括数组索引、切片、迭代、转置、变形、合并等，以及广播 (broadcasting) 机制，使得数组操作更加方便、高效。这些话题是本书后续要展开讲解的内容。

NumPy 提供了丰富的数学函数库，包括三角函数、指数函数、对数函数、逻辑函数、统计函数、随机函数等，能够满足大多数科学计算需要。



“鸢尾花书”中《数学要素》一册将大量使用这些函数库来可视化常见函数。

NumPy 支持多种文件格式的读写操作，包括文本文件、二进制文件、CSV 文件等。NumPy 基于 C 语言实现，因此可以利用底层硬件优化计算速度，同时还支持多线程、并行计算和向量化操作，使得计算更加高效。

NumPy 提供了丰富的线性代数操作方法，包括矩阵乘法、求逆矩阵、特征值分解、奇异值分解等，可以方便地解决线性代数问题。



本书中会简要介绍这些常见线性代数操作，详细讲解请大家参考“鸢尾花书”中的《矩阵力量》一册。

NumPy 可以与 `Matplotlib` 库集成使用，方便地生成各种图表，如线图、散点图、柱状图等。相信大家在本书前文已经看到基于 NumPy 数据绘制的平面、三维图像。

NumPy 提供了一些常用的数据处理方法，如排序、去重、聚合、统计等，方便对数据进行预处理。即便如此，“鸢尾花书”中我们更常用 `Pandas` 处理数据，本书后续将专门介绍 `Pandas`。

Python 中许多数据分析和机器学习的库都是基于 NumPy 创建。`Scikit-learn` 是一个流行的机器学习库，它基于 NumPy、`SciPy` 和 `Matplotlib` 创建，提供了各种机器学习算法和工具，如分类、回归、聚类、降维等。

`PyTorch` 是一个开源的机器学习框架，它基于 NumPy 创建，提供了张量计算和动态计算图等功能，可以用于构建神经网络和其他机器学习算法。

`TensorFlow` 是一个深度学习框架，它基于 NumPy 创建，提供了各种神经网络算法和工具，包括卷积神经网络、循环神经网络等。

“鸢尾花书”中的《数据有道》专门讲解回归、降维这两类机器学习算法，而《机器学习》一册则侧重分类、聚类。



本节配套的 Jupyter Notebook 文件主要是 BK_2_Topic_4.01_1.ipynb，请大家边读正文边在 JupyterLab 中探究学习。

13.2 手动构造数组

从 `numpy.array()` 说起

我们可以利用 `numpy.array()` 手动生成一维、二维、三维等数组。下面首先介绍如何使用 `numpy.array()` 这个函数。

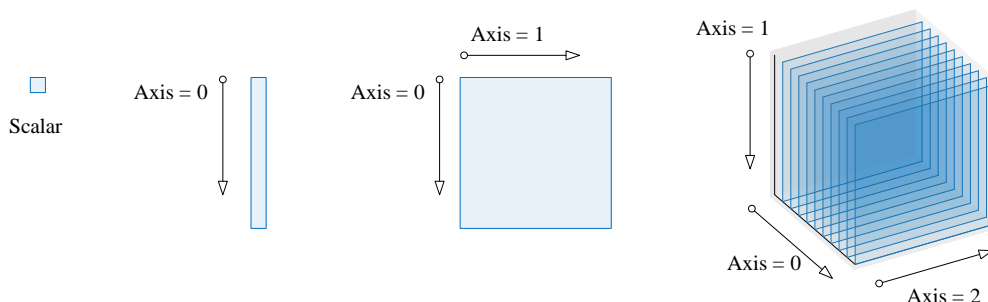


图 1. 标量、一维数组、二维数组、三维数组



`numpy.array(object, dtype)`

这个函数的重要输入参数：

- `object` 转换为数组的输入数据，可以是列表、元组、其他数组或类似序列的对象。
- `dtype` 参数用于指定数组的数据类型。如果不指定 `dtype` 参数，则 NumPy 会自动推断数组的数据类型。

请大家在 JupyterLab 中自行学习下例。

```
import numpy as np

# 从列表中创建一维数组
arr1 = np.array([1, 2, 3, 4])

# 指定数组的数据类型
arr2 = np.array([1, 2, 3, 4], dtype=float)

# 从元组中创建二维数组
arr3 = np.array([(1, 2, 3), (4, 5, 6)])
```

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

```
# 指定最小维度
arr4 = np.array([1, 2, 3, 4], ndmin=2)
```



NumPy 中的 array 是什么？

在 NumPy 中，`array` 是一种多维数组对象，它可以用于表示和操作向量、矩阵和张量等数据结构。`array` 是 NumPy 中最重要的数据结构之一，它支持高效的数值计算和广播操作，可以用于处理大规模数据集和科学计算。与 Python 中的列表不同，`array` 是一个固定类型、固定大小的数据结构，它可以支持多维数组操作和高性能数值计算。`array` 的每个元素都是相同类型的，通常是浮点数、整数或布尔值等基本数据类型。在创建 `array` 时，用户需要指定数组的维度和类型。例如，可以使用 `numpy.array()` 函数创建一个一维数组或二维数组，也可以使用 `numpy.zeros()` 函数或 `numpy.ones()` 函数创建指定大小的全 0 或全 1 数组，还可以使用 `numpy.random` 模块生成随机数组等。除了基本操作之外，NumPy 还提供了许多高级的数组操作，例如数组切片、数组索引、数组重塑、数组转置、数组拼接和分裂等。

代码 1 和代码 2 首先定义两个可视化函数。

下面，让我们首先聊聊代码 1。

- a 从 `matplotlib` 中导入 `cm` 模块。`cm` 模块提供了许多预定义的颜色映射和相关方法。
- b 自定义可视化函数用来展示二维数组。
- c 中 `array.shape[1]` 返回数组的列数，然后用 `math.ceil()` 向上取整，确保结果是整数。这个结果用来作为图像宽度。
- 类似地，d 结果用来作为图像高度。
- e 用 `matplotlib.pyplot.subplots()`，简作 `plt.subplots()`，创建图形对象 `fig` 和轴对象 `ax`。
- f 调用 `seaborn.heatmap()`，简作 `sns.heatmap()`，绘制热图可视化二维数组。`seaborn.heatmap()` 函数输入参数含义请参考代码 1 注释。

```

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
a from matplotlib import cm

# 定义二维数组可视化函数
b def visualize_2D(array, title, vmax, vmin):

c     fig_width = math.ceil(array.shape[1] * 0.5)
d     fig_length = math.ceil(array.shape[0] * 0.5)

e     fig, ax = plt.subplots(figsize=(fig_width, fig_length))

f     sns.heatmap(array,
                  vmax = vmax,
                  vmin = vmin,
                  annot = True,          # 增加注释
                  fmt = ".0f",          # 注释数值的格式
                  square = True,         # 热图方格为正方形
                  cmap = 'RdYlBu_r',    # 指定色谱
                  linewidths = .5,      # 方格线宽
                  cbar = False,          # 不显示色谱条
                  yticklabels=False,    # 不显示纵轴标签
                  xticklabels=False,    # 不显示横轴标签
                  ax = ax)              # 指定绘制热图的轴

```

代码 1. 自定义函数，可视化二维数组 | Bk1_Ch13_01.ipynb

让我们再聊聊代码 2。

- a 自定义函数用来可视化一维数组。
 - b 中首先用 `np.linspace(0, 1, len(array))` 创建一个等差数列，从 0 到 1，包含 `len(array)` 个值。然后，利用 `cm.RdYlBu_r()` 将等差数列映射到指定颜色映射上，得到一个包含 `len(array)` 个颜色值的数组 `colors`。
 - c 创建的 for 循环中，d 用 `plt.Circle()` 在指定坐标 (`idx, 0`)，绘制半径为 0.5 的圆形。参数 `facecolor` 用来指定圆形颜色，参数 `edgecolor` 指定圆形边缘颜色为白色。
 - e 用 `add_patch()` 方法在轴对象上添加圆形；注意，这一步不可以省去，不然无法显示圆形对象。
 - f 用 `text()` 在指定位置显示数组中索引为 `idx` 的数值。`horizontalalignment='center'` 和 `verticalalignment='center'` 分别设置文本对象在水平和垂直方向上的对齐方式为居中。
 - g 设置横轴、纵轴比例尺相同。
 - h 隐藏坐标值。
- ➔ 鸢尾花书《可视之美》会专门介绍如何用 Matplotlib 绘制各种几何图形。

```


# 定义一维数组可视化函数
a def visualize_1D(array, title):
    fig, ax = plt.subplots()

    b colors = cm.RdYlBu_r(np.linspace(0,1,len(array)))

    c for idx in range(len(array)):
        d circle_idx = plt.Circle((idx, 0), 0.5,
                                   facecolor=colors[idx],
                                   edgecolor = 'w')
        e ax.add_patch(circle_idx)
        f ax.text(idx, 0, s = str(array[idx]),
                  horizontalalignment = 'center',
                  verticalalignment = 'center')

    ax.set_xlim(-0.6, 0.6 + len(array))
    ax.set_ylim(-0.6, 0.6)
    g ax.set_aspect('equal', adjustable='box')
    h ax.axis('off')

```

代码 2. 自定义函数，可视化一维数组 |  Bk1_Ch13_01.ipynb

手动生成一维数组

在 NumPy 中，一维数组是最基本的数组类型。顾名思义，一维数组只有一个维度，可以包含多个元素，一般数组中每个元素具有相同数据类型。


图 2 所示为利用 `numpy.array()` 生成的一维数组。这个数组的形状为 `(7,)`，长度为 7，维度为 1。

和本书前文介绍的 `list` 一样，NumPy 数组的索引也是从 0 开始。下一话题专门讲解 NumPy 数组索引和切片。再次强调，如图 2 所示，本书可视化一维数组时一般用圆形。

```

a = numpy.array([-3, -2, -1, 0, 1, 2, 3])
axis = 0 →

```



Index	0	1	2	3	4	5	6
Value	-3	-2	-1	0	1	2	3

图 2. 手动生成一维数组 |  Bk1_Ch13_01.ipynb

请大家自行学习代码 3，并逐行注释。

```

# 定义一维数组
a a_1D = np.array([-3, -2, -1, 0, 1, 2, 3])
  print(a_1D)
b print(a_1D.shape)
c print(len(a_1D))
d print(a_1D.ndim)
e print(a_1D.size)
# 可视化
f visualize_1D(a_1D, '手动, 一维')

```

代码 3. 一维 NumPy 数组, 使用时配合前文代码 | Bk1_Ch13_01.ipynb

下面区分一下形状、长度、维度、大小这四个特征：

- ▶ 形状：可以使用 `shape` 属性来获取数组的形状；如果 `arr` 是一个二维数组，则可以使用 `arr.shape` 来获取其形状，行、列数。
- ▶ 长度：可以使用 `len()` 函数来获取数组的长度；如果 `arr` 是一个一维数组，则可以使用 `len(arr)` 来获取其长度，即元素数量；如果 `arr` 是个二维数组，`len(arr)` 返回行数。
- ▶ 维数：可以使用 `ndim` 属性来获取数组的维数；如果 `arr` 是一个二维数组，则可以使用 `arr.ndim` 来获取其维数，即 2。
- ▶ 大小：可以使用 `size` 属性来获取数组所有元素的个数；如果 `arr` 是一个二维数组，则可以使用 `arr.size` 来获取所有元素个数。

手动生成二维数组

图 3 所示为利用 `numpy.array()` 生成的二维数组。利用 `ndim` 方法，大家可以发现图 3 中数组的维度都是 2。此外，`numpy.matrix()` 专门用来生成二维矩阵，请大家自行学习。

⚠ 请大家注意图 3 中括号 `[]` 的数量。特别强调，本书中，行向量、列向量都被视作特殊的二维数组。可以这样理解，行向量是一行多列矩阵，而列向量是多行一列矩阵。

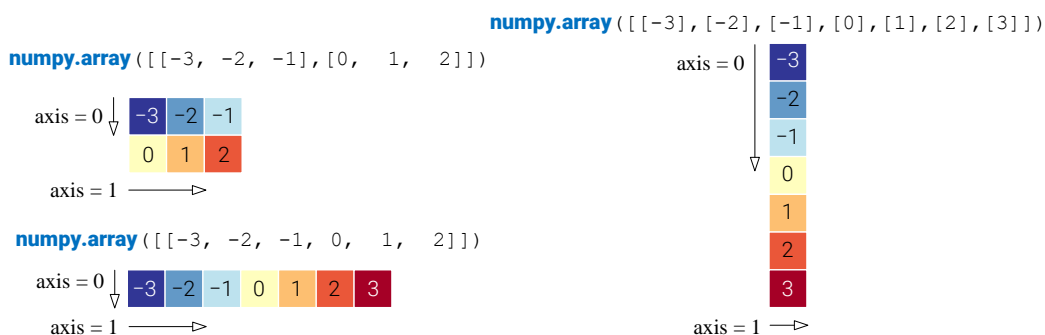




图 3. 手动生成二维数组 | Bk1_Ch13_01.ipynb

请大家自行学习代码 4、代码 5、代码 6，并逐行注释。


```
# 定义二维数组
a a_2D = np.array([[ -3, -2, -1],
                  [ 0,  1,  2]])
print(a_2D)
# 可视化
b visualize_2D(a_2D, '手动, 二维', 3, -3)
c print(a_2D.shape)
d print(a_2D.shape[0]) # 行数
e print(a_2D.shape[1]) # 列数
f print(a_2D.ndim)
g print(a_2D.size)
h print(len(a_2D))
```

代码 4. 二维 NumPy 数组，形状为 (2, 3)，使用时配合前文代码 |  Bk1_Ch13_01.ipynb

```
# 定义二维数组，行向量（两层中括号）
a a_row_vector = np.array([[ -3, -2, -1, 0, 1, 2, 3]])
# 可视化
b visualize_2D(a_row_vector, '手动, 行向量', 3, -3)
print(a_row_vector.shape)
print(a_row_vector.ndim)
```

代码 5. 二维 NumPy 数组，形状为 (1, 7)，使用时配合前文代码 |  Bk1_Ch13_01.ipynb

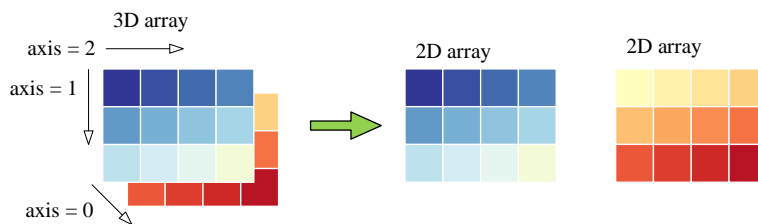
```
# 定义二维数组，列向量
a a_col_vector = np.array([[ -3], [-2], [-1], [0], [1], [2], [3]])
# 可视化
b visualize_2D(a_col_vector, '手动, 列向量', 3, -3)
print(a_col_vector.shape)
print(a_col_vector.ndim)
```

代码 6. 二维 NumPy 数组，形状为 (7, 1)，使用时配合前文代码 |  Bk1_Ch13_01.ipynb

手动生成三维数组

图 4 所示为利用 `numpy.array()` 生成的三维数组，这个数组的形状为 (2, 3, 4)，也就是 2 页 (`axis = 0`)、3 行 (`axis = 1`)、4 列 (`axis = 2`)。


本章配套 Bk1_Ch13_01.ipynb 展示如何获取三维数组的第 0 页和第 1 页。

图 4. 手动生成三维数组 |  Bk1_Ch13_01.ipynb

请大家自行学习代码 7，请逐行注释。

```
# 定义三维数组
a a_3D = np.array([[[[-12, -11, -10, -9],
                    [-8, -7, -6, -5],
                    [-4, -3, -2, -1]],
                   [[0, 1, 2, 3],
                    [4, 5, 6, 7],
                    [8, 9, 10, 11]]]])

b print(a_3D.shape)
c print(a_3D.ndim)
# 可视化
d visualize_2D(a_3D[0], '手动, 三维, 第一页', 12, -12)
e print(a_3D[0].shape)
f visualize_2D(a_3D[1], '手动, 三维, 第二页', 12, -12)
```


代码 7. 三维 NumPy 数组，形状为 (2, 3, 4)，使用时配合前文代码 |  Bk1_Ch13_01.ipynb

我们也可以用 `numpy.array()` 将列表 `list` 转化为 NumPy 数组，代码 8 给出三个示例，请自行学习并逐行注释。请大家格外注意中括号 `[]` 层数。

```
# 一维数组
a list_1D = [-3, -2, -1, 0, 1, 2, 3]
array_1D = np.array(list_1D)
print(array_1D.shape)

# 二维数组
b list_2D = [[-3, -2, -1, 0, 1, 2, 3]]
array_2D = np.array(list_2D)
print(array_2D.shape)

# 三维数组
c list_3D = [[[ -3, -2, -1, 0, 1, 2, 3 ]]]
array_3D = np.array(list_3D)
print(array_3D.shape)
```

代码 8. 将列表 list 转化为 NumPy 数组 |  Bk1_Ch13_01.ipynb

13.3 生成数列

在 NumPy 中我们常用以下三个函数生成数列（一维数组）。

- ▶ `numpy.arange(start, stop, step)` 生成等差数列；从起始值 `start` 开始，以步长 `step` 递增，直到结束值 `stop`（不包含 `stop`）。例如，`numpy.arange(1, 11, 2)` 生成等差数列 `[1, 3, 5, 7, 9]`。实际上，`numpy.arange()` 和前文介绍的 `range()` 函数颇为相似。
- ▶ `numpy.linspace(start, stop, num, endpoint)` 生成等差数列；从起始值 `start` 开始，到结束值 `stop` 结束，`num` 指定数列的长度（元素的个数），默认为 50。`endpoint` 参数指定是否包含结束值。例如，`numpy.linspace(1, 10, 5)` 生成等差数列 `[1, 3.25, 5.5, 7.75, 10]`。
- ▶ `numpy.logspace(start, stop, num, endpoint, base)` 生成等比数列；从 `base` 的 `start` 次幂开始，到 `base` 的 `stop` 次幂结束，`num` 指定数列的长度，默认为 50。例如，`numpy.logspace(0, 4, 5, base=2)` 将生成一个等比数列 `[1, 2, 4, 8, 16]`。

请大家在 JupyterLab 中自行练习表 1 中几个例子。



什么是数列？

数列是指一系列按照一定规律排列的数，它通常用一个公式来表示，也可以用递推关系式来定义。数列中的每个数称为数列的项，用 a_n 来表示第 n 项。数列在数学中具有广泛的应用，它是许多数学分支的基础，如数学分析、概率论、统计学、离散数学和计算机科学等。在数学中，数列是一种有序的集合，通常用于研究数学对象的性质和行为，例如函数、级数、微积分和代数等。数列可以分为等差数列、等比数列和通项公式不规则数列等几种类型。等差数列的项之间的差是固定的，比如 1、2、3、4 ... 100。等比数列的相邻项之间的比是固定的，比如 2、4、8、16 ... 2048。

表 1. 生成数列

代码示例	结果
<code>import numpy as np</code> <code>np.arange(5)</code>	<code>array([0, 1, 2, 3, 4])</code>
<code>np.arange(5, dtype = float)</code>	<code>array([0., 1., 2., 3., 4.])</code>
<code>np.arange(10, 20)</code>	<code>array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])</code>
<code>np.arange(10, 20, 2)</code>	<code>array([10, 12, 14, 16, 18])</code>
<code>np.arange(10, 20, 2, dtype = float)</code>	<code>array([10., 12., 14., 16., 18.])</code>
<code>np.linspace(0, 5, 11)</code>	<code>array([0., 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.])</code>
<code>np.logspace(0, 4, 5, base=10)</code>	<code>array([1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04])</code>
<code>np.logspace(0, 4, 5, base=2)</code>	<code>array([1., 2., 4., 8., 16.])</code>

13.4 生成网格数据

本书前文提过 `numpy.meshgrid()` 函数。我们还自己写代码复刻这个函数结果。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

简单来说，`numpy.meshgrid()` 可以生成多维网格数据，它可以将多个一维数组组合成一个 N 维数组，并且可以方便地对这个 N 维数组进行计算和可视化。

在科学计算中，常常需要对多维数据进行可视化，比如绘制 3D 曲面图、等高线图等。`numpy.meshgrid()` 可以方便地生成网格坐标。

对于二元函数 $f(x_1, x_2)$ ，我们可以使用 `numpy.meshgrid()` 生成横坐标和纵坐标的网格点，然后计算每个网格点的函数值。最后将网格坐标 (`xx1`, `xx2`) 和对应的函数值 (`ff`) 作为输入，绘制出如图 5 所示的 3D 曲面图。

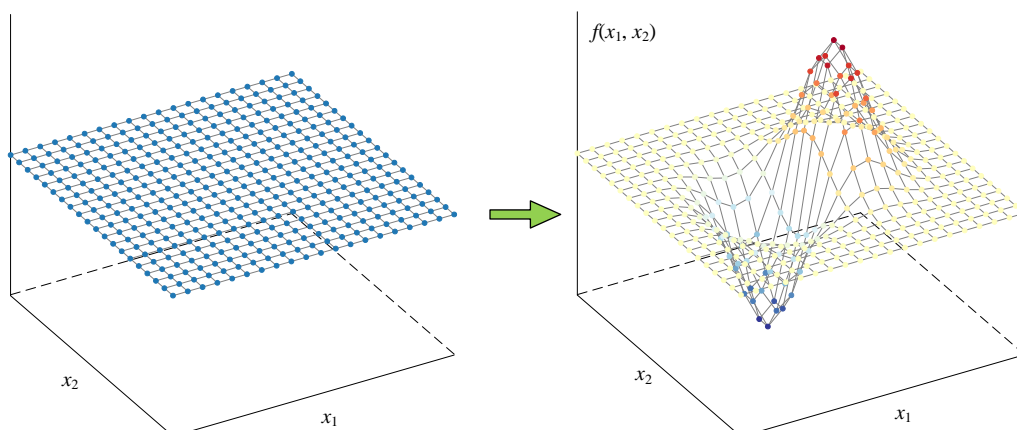


图 5. 三维空间看二维网络状坐标

代码 9 绘制图 5 右图，下面聊聊其中关键语句。

- a 利用 `numpy.meshgrid()` 生成网格化数据（二维数组），代表 x 和 y 轴坐标。
- b 计算二元函数 $f(x_1, x_2) = x_1 \exp(-x_1^2 - x_2^2)$ 函数值，结果为二维数组。
- c 用 `matplotlib.pyplot.figure()`，简作 `plt.figure()`，创建图形对象 `fig`。
- d 在图像对象 `fig` 上，用 `add_subplot()` 方法增加三维轴对象 `ax`。
- e 在 `ax` 上用 `plot_wireframe()` 绘制三维网格图。
- f 在 `ax` 上用 `scatter()` 绘制三维散点图。函数值 `ff` 大小作为参考用颜色映射 `RdYlBu_r` 渲染散点。

```

import numpy as np
import matplotlib.pyplot as plt

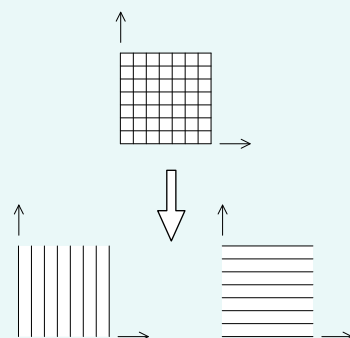
x1_array = np.linspace(-3, 3, 21)
x2_array = np.linspace(-3, 3, 21)

a xx1, xx2 = np.meshgrid(x1_array, x2_array)
  # 二元函数
b ff = xx1 * np.exp(-xx1**2 - xx2**2)
  print(xx1.shape)

  # 可视化
c fig = plt.figure()
d ax = fig.add_subplot(projection='3d')

e ax.plot_wireframe(xx1, xx2, ff,
                    rstride=1, cstride=1,
                    color = 'grey')
f ax.scatter(xx1, xx2, ff, c = ff, cmap = 'RdYlBu_r')
  ax.set_proj_type('ortho')
  plt.show()

```


代码 9. 可视化二元函数 |  Bk1_Ch13_01.ipynb

如图 6 所示，`numpy.meshgrid()` 还可以用来生成三维网格数据。



在《可视之美》一册中，大家可以看到大量利用三维网格数据完成的可视化方案。

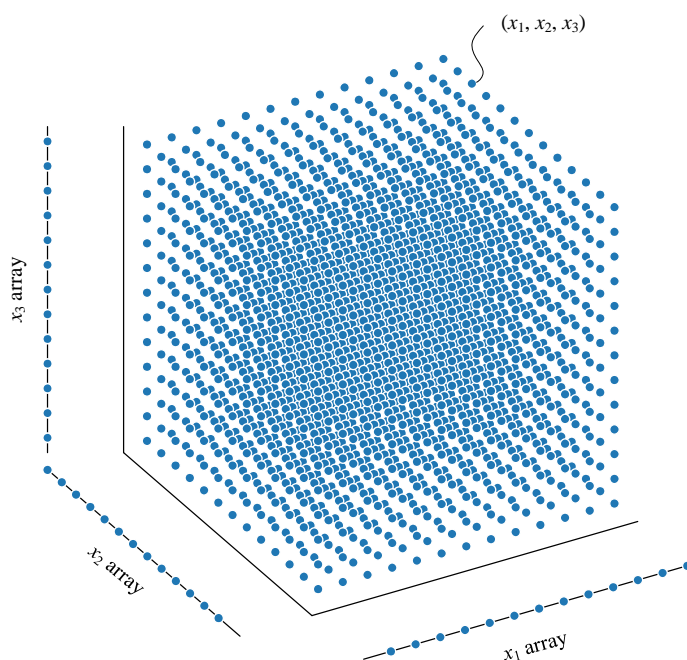


图 6. 三维网格

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

13.5 特殊数组

表 2 总结 NumPy 中常用来生成特殊数组的函数、用途、示例，请大家在 JupyterLab 中练习使用这些函数。

表 2. 用 NumPy 函数生成特殊数组

函数	用途	代码示例
<code>numpy.empty()</code>	创建一个指定大小、未初始化的数组	<pre>import numpy as np np.empty([4,4])</pre>
<code>numpy.empty_like()</code>	创建与给定数组具有相同形状和数据类型的未初始化数组的函数	<pre>A = np.array([[1, 2, 3], [4, 5, 6]]) np.empty_like(A)</pre>
<code>numpy.eye()</code>	创建一个二维数组，表示单位矩阵	<pre>np.eye(5)</pre>
<code>numpy.full()</code>	创建一个指定大小和给定值的数组	<pre>np.full((3,3), np.inf)</pre>
<code>numpy.full_like()</code>	创建与给定数组具有相同形状和数据类型，且所有元素都是指定值	<pre>A = np.array([[1, 2, 3], [4, 5, 6]]) np.full_like(A, 100)</pre>
<code>numpy.ones()</code>	创建一个指定大小的全 1 数组	<pre>np.ones((5,5))</pre>
<code>numpy.ones_like()</code>	创建与给定数组具有相同形状和数据类型，且所有元素都是 1 的数组	<pre>A = np.array([[1, 2, 3], [4, 5, 6]]) np.ones_like(A)</pre>
<code>numpy.zeros()</code>	创建一个指定大小的全 0 数组	<pre>np.zeros((5,5))</pre>
<code>numpy.zeros_like()</code>	创建与给定数组具有相同形状和数据类型，且所有元素都是 0 的数组	<pre>A = np.array([[1, 2, 3], [4, 5, 6]]) np.zeros_like(A)</pre>



什么是单位矩阵？

单位矩阵是一个非常特殊的方阵，它的对角线上的元素全都是 1，而其余元素全都是 0。常用符号表示单位矩阵的是 I 或者 E ，它的大小由下标表示，例如， I_2 表示 2×2 的单位矩阵。类似地， I_3 表示 3×3 的单位矩阵，以此类推。单位矩阵是在矩阵运算中非常重要的一个概念，它可以被看作是矩阵乘法中的“1”，即任何矩阵与单位矩阵相乘，其结果都是该矩阵本身。单位矩阵在许多应用中都有广泛的应用，例如，单位矩阵可以用来表示标准正交基等。在计算矩阵的逆时，单位矩阵也起到了关键作用，因为一个矩阵 A 的逆矩阵可以通过 A 和单位矩阵的运算来计算，即 $AA^{-1} = A^{-1}A = I$ 。

13.6 随机数

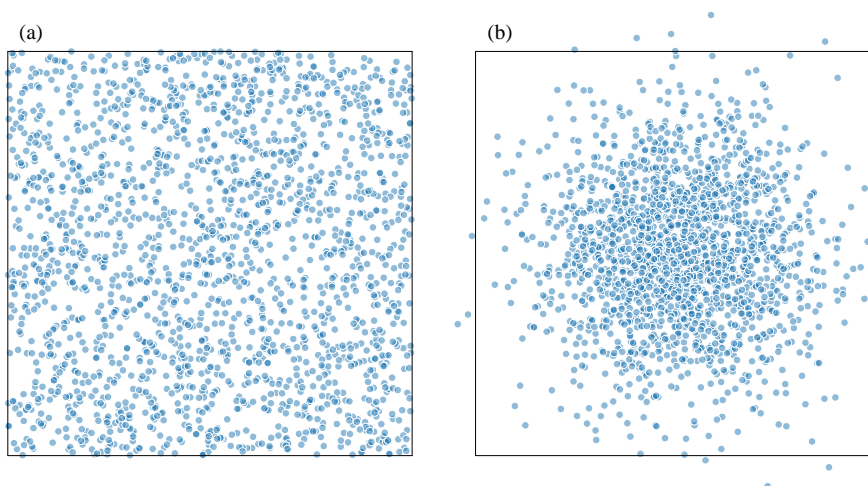

NumPy 中还有大量产生随机数的函数。图 7 所示为满足二元连续均匀分布、二元高斯分布的随机数。

代码 10 绘制图 7 (a)，代码 11 绘制图 7 (b)。请大家翻阅帮助文档了解这两段代码中主要函数的用法，并在 JupyterLab 中动手实践。

表 2 总结 NumPy 中常用随机数发生器函数和随机数分布图像。



“鸢尾花书”《统计至简》一册将专门讲解各种常用概率分布。


图 7. 分别满足二元连续均匀分布、二元高斯分布的随机数 |  Bk1_Ch13_01.ipynb

```
import numpy as np
import matplotlib.pyplot as plt

# 生成随机数，服从连续均匀分布
num = 2000
a X_uniform = np.random.uniform(low=-3, high=3, size=(num,2))

fig, ax = plt.subplots(figsize = (5,5))
b ax.scatter(X_uniform[:,0], # 散点横轴坐标
            X_uniform[:,1], # 散点纵轴坐标
            s = 100,         # 散点大小
            marker = '.',    # 散点marker样式
            alpha = 0.5,     # 透明度
            edgecolors = 'w') # 散点边缘颜色

ax.set_aspect('equal', adjustable='box')
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.set_xticks((-3,0,3))
ax.set_yticks((-3,0,3))
```

代码 10. 服从连续均匀的随机数 |  Bk1_Ch13_01.ipynb

```

import numpy as np
import matplotlib.pyplot as plt

# 生成随机数，服从二元高斯分布
num = 2000

a mu = np.array([0, 0])      # 质心
b rho = 0 # 相关性系数
c Sigma = np.array([[1, rho],
                    [rho, 1]]) # 协方差矩阵

d X_binormal = np.random.multivariate_normal(mu, Sigma, size=num)

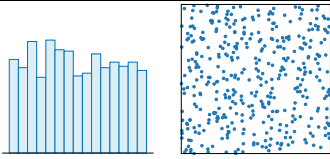
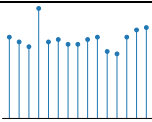
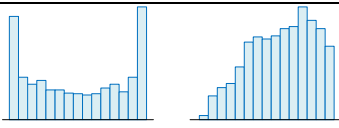
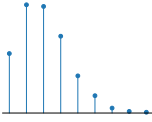
fig, ax = plt.subplots(figsize = (5,5))
ax.scatter(X_binormal[:,0],
           X_binormal[:,1],
           s = 100,
           marker = '.',
           alpha = 0.5,
           edgecolors = 'w')

ax.set_aspect('equal', adjustable='box')
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.set_xticks((-3,0,3))
ax.set_yticks((-3,0,3))

```

代码 11. 服从二元高斯分布随机数 | Bk1_Ch13_01.ipynb

表 3. 常用随机数发生器

随机数服从的分布	函数	随机数分布图像
连续均匀分布	<code>numpy.random.uniform()</code>	
均匀整数	<code>numpy.random.randint()</code>	
Beta 分布	<code>numpy.random.beta()</code>	
泊松分布	<code>numpy.random.poisson()</code>	

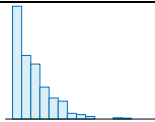
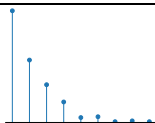
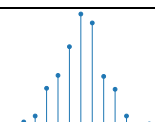
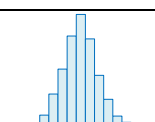
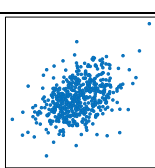
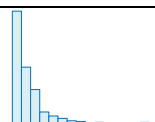
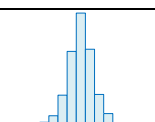
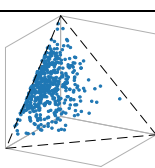
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

指数分布	<code>numpy.random.exponential()</code>	
几何分布	<code>numpy.random.geometric()</code>	
二项分布	<code>numpy.random.binomial()</code>	
正态分布	<code>numpy.random.normal()</code>	
多元正态分布	<code>numpy.random.multivariate_normal()</code>	
对数正态分布	<code>numpy.random.lognormal()</code>	
学生 t-分布	<code>numpy.random.standard_t()</code>	
Dirichlet 分布	<code>numpy.random.dirichlet()</code>	



概率统计中，随机是什么意思？

在概率统计中，随机指的是一个事件的结果是不确定的，而且每种可能的结果出现的概率是可以计算的。随机事件是由各种随机变量所描述的，随机变量是一个具有不确定结果的数学变量，其值取决于随机事件的结果。概率统计学家使用随机变量和概率分布来描述随机事件的结果和出现的概率。随机事件的结果可能是离散的，例如掷骰子的结果是 1、2、3、4、5 或 6，也可能是连续的，例如衡量人的身高或重量。概率统计学家使用各种数学方法和技术，例如概率、期望值和方差等，来分析和理解随机事件和随机变量的性质和行为。概率统计的研究在现代科学和工程中有着广泛的应用，例如金融、生物学、医学、物理学等领域。



什么是随机数发生器？

随机数生成器是一种用于生成随机数的计算机程序或硬件设备。随机数生成器可分为真随机数生成器和伪随机数生成器两种。真随机数生成器的输出完全基于物理过程，如大气噪声、放射性衰变或者热噪声等，其生成的随机数序列是完全随机且不可预测的。真随机数生成器通常需要专门的硬件设备支持。伪随机数生成器则使用计算机算法生成伪随机数，其看似随机，但是实际上是可预测的，因为它们是由固定的算法和种子值生成的。伪随机数生成器通常使用伪随机数序列和随机种子，以便在需要时生成

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

随机数。随机数生成器在计算机科学、加密学、模拟实验、游戏设计、统计分析等领域中被广泛使用。在加密学中，随机数生成器通常用于生成安全密钥和初始化向量等关键数据，以保证加密算法的强度和安全性。在模拟实验和游戏设计中，随机数生成器用于模拟不可预测的因素，如掷骰子、扑克牌等。

13.7 数组导入、导出

图 8 所示为鸢尾花表格和热图。

代码 12 绘制图 8，下面聊聊其中关键语句。

a 用 `sklearn.datasets` 模块中 `load_iris()` 函数导入鸢尾花数据，存为 `iris.iris.data` 保存鸢尾花特征数据，形式为 NumPy Array。

b 用 `numpy.savetxt()` 把 numpy array 写成 CSV 文件。CSV (comma-separated values)，即逗号分隔值文件，是一种常见的文本文件格式，用于存储表格数据。CSV 文件中的数据以纯文本形式表示，通常使用逗号来分隔不同的字段或列，而行则用换行符分隔。

c 用 `numpy.genfromtxt()` 读入 CSV 文件。

d 用 `seaborn.heatmap()` 可视化鸢尾花特征数据矩阵。

大家在本书后文，特别是在鸢尾花书《矩阵力量》一册中会看到，我们大量使用热图可视化矩阵运算。

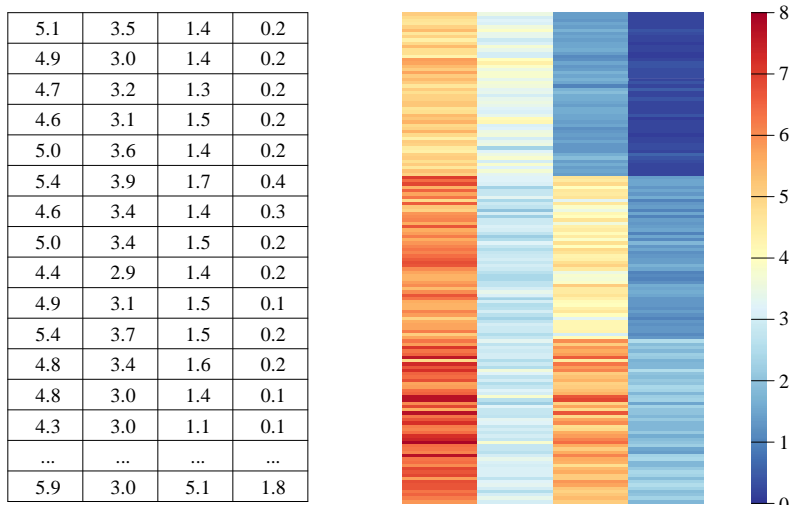


图 8. 鸢尾花数据表格和热图 |  Bk1_Ch13_01.ipynb

```

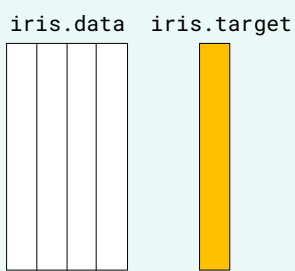
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from numpy import genfromtxt

# 导入鸢尾花数据
a iris = load_iris()
# 将numpy array存成CSV文件
b np.savetxt("Iris_data.csv", iris.data, delimiter=",")

# 将 CSV 文件读入存成numpy array
c Iris_Data_array = genfromtxt('Iris_data.csv', delimiter=',')

# 可视化
fig, ax = plt.subplots(figsize = (5,5))
d sns.heatmap(Iris_Data_array, # 鸢尾花数据数组
              cmap = 'RdYlBu_r', # 指定色谱
              ax = ax, # 指定轴
              vmax = 8, # 色谱最大值
              vmin = 0, # 色谱最小值
              xticklabels = [], # 不显示横轴标签
              yticklabels = [], # 不显示纵轴标签
              cbar = True) # 显示色谱条

```



代码 12. 用热图可视化鸢尾花数据 | Bk1_Ch13_01.ipynb



请大家完成下面 3 道题目。

Q1. 用至少两种办法生成一个 3×4 二维 NumPy 数组，数组的每个值都是 10。

Q2. 利用 `numpy.meshgrid()` 和 `matplotlib.pyplot.contour()` 绘制二元函数 $f(x_1, x_2) = x_2 \exp(-x_1^2 - x_2^2)$ 的平面等高线。

Q3. 在 $[0, 1]$ 范围内生成 1000 个满足连续均匀随机数，并用 `matplotlib.pyplot.hist()` 绘制频率直方图。

* 题目答案在 Bk1_Ch13_02.ipynb。



NumPy 最大的优势在于提供了高性能的多维数组对象和相应的操作函数，使得矩阵相关计算更加高效。在机器学习中，NumPy 常用于数据处理、线性代数运算和数组操作，为模型训练提供了基础。

本书前文介绍过用嵌套列表代表矩阵，本章开始请大家利用 NumPy Array。大家会发现 NumPy Array 的向量化运算特别方便，帮助我们避免了很多循环。